Safer Copying

Processing User Input

BY NEIL J. RUBENKING

SAFER COPYING

Many DOS commands permit the use of wildcards to perform an operation on a set of files or directories. But if you aren't careful, this can have unexpected results. Have you ever copied the contents of one directory to another, inadvertently mistyping the destination? You don't get what you wanted, and worse yet, you get no error message. Here's a simple tip.

Suppose you want to copy all files in directory X to directory Y on the A: drive. What would happen if you type the command `COPY X A:Y`, and directory Y doesn't exist? The A: drive will contain only a new file called Y, and you won't get an error message.

That's not good. The following variation does produce an error if directory Y does not exist on the A: drive.

```
COPY X A:Y\*.*
```

Better still, use this command to protect against mistyping the source directory name:

```
COPY X\*.* A:Y\*.*
```

But typing *.* is a bit of a bother. I've discovered that the period is accepted by many DOS commands as a synonym for *.*, so I've been able to use

```
COPY X\. A:Y\.
```

I haven't yet encountered a version of DOS that won't accept a period for commands like COPY, DIR, and PRINT. One notable exception is the FOR command's list, in which a period is *not* the same as *.*.

I routinely append a backslash and period to any directory argument where the directory should already exist. This has saved me numerous times from failing to copy what I wanted onto backup floppy disks and directories. Typing \. is about as easy as it gets, since it doesn't require the Shift key.

Fred C. Belsak
North Chelmsford, Massachusetts

▶PC MAGAZINE: DOS's COPY command is a lot more versatile than most users realize, and it's one of COPY's more obscure features that causes the problem noted above. COPY has the ability to concatenate files: The command `COPY A+B C` will create a new file, C, containing the contents of both file A and file B. When you COPY from a wildcard file specification to a single filename, COPY will concatenate all matching files and store the result in the target file. That's exactly what happens in the example `COPY X A:Y`. COPY treats the directory name X as equivalent to X*.*, so it copies all files in the directory X to the single file Y on the A: drive.

But wait—It gets worse! By default, COPY uses *binary* mode, in which it copies every byte from source to target. But when concatenating files, COPY uses *ASCII* mode, in which the target is truncated at the first Ctrl-Z character. So the single file Y most likely does not contain all the data from the files in directory X.

In the solution offered, the period doesn't actually substitute for *.*. Rather, it is a substitute for a directory name. Every subdirectory contains a directory entry whose name consists of a single period; this entry refers to the directory itself. In a DOS command, a lone period refers to the current directory, and a backslash and period following a directory name refers to that directory. In fact, you can keep appending backslash-period pairs to a directory name indefinitely (or until you run into the 127-character command-line limit). Try it! You will see that `DIR C:\.\.\.\.\.` yields precisely the same result as `DIR C:\`.

Adding a backslash and period to a directory name effectively leaves it unchanged, but the same is *not* true of filenames. By appending this pair of characters every time you use the name of a directory that should exist, you force DOS commands to halt with an error message if you mistype the directory name. That's better than finding out later that your archives are incomplete!

PROCESSING USER INPUT

I've found another way to give a batch file access to an input string. My version consists of two files, INP.COM and INPUT.BAT. INP.COM is a small program you can create with the DEBUG script in Figure 1, using the command

```
DEBUG < INP.SCR
```

The program waits for string input from a user, then creates a file named TEMP.BAT containing that input. INP.COM is a very simple program and no error checking is done after each DOS function call.

INPUT.BAT, shown below, uses INP.COM to assign an input string to an

INPUT.SCR
Complete Listing

```
N INP.COM
E 144 "TEMP.BAT",0
A 100
MOV     DX,0144                 ; file name
MOV     AX,3C00
MOV     CX,0
INT     21                      ; create file
PUSH    AX                      ; file handle in stack
MOV     CH,00
MOV     CL,[080]                ; length of command string
JCXZ    11F                     ; skip if no command line
DEC     CL                      ; drop first char (space)
POP     BX                      ; get file handle
PUSH    BX
MOV     AH,40
MOV     DX,082                  ; command string offset
INT     21                      ; write to file
MOV     AH,0A
MOV     DX,0144
MOV     BYTE PTR [0144],50      ; max. length of 80 characters
INT     21                      ; input string
MOV     CH,00
MOV     CL,[0145]               ; length of string
POP     BX                      ; get file handle
PUSH    BX
MOV     AH,40
MOV     DX,0146                 ; input string offset
INT     21                      ; write to file
POP     BX
MOV     AH,3E
INT     21                      ; close file
MOV     AX,4C00
INT     21                      ; exit to dos

R CX
4D
W
Q
```

Figure 1: **This DEBUG script defines the file INP.COM, which makes it easy to get input from a user in a batch file.**

environment variable:

```
@ECHO OFF
ECHO Enter a value for environment
 variable %1:
INP SET %1=
CALL  TEMP.BAT
DEL   TEMP.BAT
```

Here's the syntax you should use: `INPUT <variable name>`. You can issue this instruction either from the DOS command line or within a batch file. Either way, don't forget that you will need to use the CALL command to execute INPUT from another batch file.

The following batch file, SAVEDIR.BAT, illustrates another way to use INP.COM:

```
@ECHO OFF
CD¦INP SET DIR=>NUL
CALL TEMP.BAT
DEL  TEMP.BAT
```

SAVEDIR saves the name of the current directory into an environment variable named DIR.

Nuno Macedo Silva
Porto, Portugal

▶PC MAGAZINE: We've presented a number of ways for a batch file to obtain string input from a user, but INP.COM may be the easiest yet. The ability to process a line of input from the user is essential to many batch files. But INP.COM can also process the output of other programs and commands, as the SAVEDIR.BAT example shows.

Sometimes you'll want to deal with just part of a line of input. Rather than set an environment variable to the line, just have TEMP.BAT call another batch file. For example, the batch file FILEBYTE.BAT shown in Figure 2 gets the number of bytes occupied by a particular group of files and stores that number in an environment variable. It pipes the output of the DIR command through the FIND filter, selecting the line that contains the word `file(s)`, and this line is piped as input to INP.COM. INP.COM writes the string

```
CALL $FILEBYT.BAT
```

to the batch file TEMP.BAT, followed by the filtered output of the DIR command.

TEMP.BAT now contains a line like this:

```
CALL $FILEBYT.BAT 2 file(s)
 1393 bytes
```

The next line of FILEBYTE.BAT creates the temporary batch file $FILEBYT.BAT, whose single line sets the e-var *filebytes* to the value of its third parameter. FILEBYTE.BAT executes TEMP.BAT, which in turn executes $FILEBYT.BAT. Then it deletes the two temporary files and reports the number of bytes occupied by the file specification. The number remains available in the e-var filebytes.

You can modify this technique to extract data from the output of any program that writes to DOS standard output. The steps will be the same. First, you should use FIND to filter out everything but the line that contains the desired information. Pipe the output of FIND into INP.COM, and put the name of a temporary batch file on INP.COM's command line. Create the temporary batch file so that it sets an environment variable to the appropriate command-line parameter. Finally CALL TEMP.BAT and delete both temporary batch files. The data is now in an e-var!

You can download all of the files that are mentioned here from the Utilities/Tips Forum on PC MagNet, archived as INPUT.ZIP. □

Share your latest DOS and system discoveries through User-to-User. See the "How to Contact Us" sidebar in the Solutions column.

FILEBYTE.BAT
Complete Listing

```
@ECHO OFF
REM FILEBYTE.BAT - Pass a file specification and
REM    this batch file will tell you how many bytes
REM    it occupies.
SET filebytes=0
IF '%1'=='' GOTO Report
IF NOT EXIST %1 GOTO Report
DIR %1 ¦ FIND "file(s)" ¦ INP CALL $FILEBYT.BAT>NUL
ECHO SET filebytes=%%3>$FILEBYT.BAT
CALL TEMP.BAT
DEL  TEMP.BAT
DEL  $FILEBYT.BAT
:===== Report =====
ECHO %filebytes% bytes occupied by "%1"
```

Figure 2: **FILEBYTE.BAT further processes input received by INP.COM, extracting a single number from the input line.**